# Qualitative behavioral reasoning from components' interfaces to components' functions for DMU adaption to FE analyses

Ahmad Shahwan, Jean-Claude Léon, Gilles Foucault, Moreno Trlin, Olivier Palombi

# Qualitative behavioral reasoning from components' interfaces to components' functions for DMU adaption to FE analyses

Ahmad Shahwan [a,*] Jean-Claude Léon [b] Gilles Foucault [a] Moreno Trlin [a] Olivier Palombi [b]

[a] *University of Grenoble - G-SCOP laboratory, 46 av. Félix Viallet, 38000, Grenoble, France*

[b] *University of Grenoble / INRIA - JLK laboratory, 655, av. de l'Europe, 38334 Montbonnot, France*

**Abstract**

A Digital Mock-Up (DMU), with its B-Rep model of product components, is a standard industrial representation that lacks geometric information about interfaces between components. Component shapes reflect common engineers practice that influence component interfaces with interferences and not only contacts.

The proposed approach builds upon relationships between function, behavior and shape to derive functional information from the geometry of component interfaces. Among these concepts, the concept of behavior is more difficult to set up and connect to the geometry of interfaces and functions. Indeed, states and design rules are introduced to express the behavior of components through a qualitative reasoning process. This reasoning process, in turn, takes advantage of domain knowledge rules and facts, checking the validity of certain hypotheses that must hold true all along a specific state of the product's lifecycle, such as operational, stand-by or relaxed states. Eliminating configurations that contradict one or more of those hypotheses in their corresponding reference state reduces ambiguity, subsequently producing functional information in a bottom-up manner.

This bottom-up process starts with the generation of a Conventional Interfaces Graph (CIG) with components as nodes, and conventional interfaces (CI) as arcs. A CI is initially defined by a geometric interaction that can be a contact or an interference between two components. CIs are then populated with Functional Interpretations (FI) according to their geometric properties, producing potentially many combinations. A first step of the reasoning process, the validation against reference states, reduces the number of FIs per CI.

Domain knowledge rules are then applied again to group semantics of components interfaces into one functional designation per component to connect together geometric entities of its boundary with its function.

*Key words:* product design, DMUs, geometric model, functional designation, semantics, reasoning and knowledge management

## 1. Introduction

Generating models for Finite Element (FE) analysis from B-Rep CAD models of components often requires shape transformations to remove details [1,2] or idealize sub-domains [3] to meet the engineer's requirements in terms of FE mesh generation, simulation objectives, accuracy and time to fit into a Product Development Process (PDP) [4]. The corresponding geometric transformations are still raising open issues to achieve efficiently these transformations under simulation criteria. However, companies, e.g. at the aircraft industry, are starting to increase the complexity of simulation models where the objective is to simulate the structural behavior of assemblies with up to thousands of components. In this case, the starting point of the FE preparation process is a Digital Mock-Up (DMU), as available from CAD software. Currently, this task is very tedious for a small assembly with tens of components and it is not achievable for very large ones because most of the processing is performed interactively by structural engineers. Indeed, a DMU contains essentially geometric entities that can support this preparation process whereas engineers refer to component functions to decide whether and how they can be idealized (see Fig. 2), e.g. whether bolts should be idealized as beams or simplified as shapes of revolution with friction areas set up at the interfaces between plates. To this end, shape transformations can be automated if component functions are

* tel. +33 608552947

*Email addresses:* Ahmad.Shahwan@grenoble-inp.fr (Ahmad Shahwan), Jean-Claude.Leon@grenoble-inp.fr (Jean-Claude Léon), Gilles.Foucault@ujf-grenoble.fr (Gilles Foucault), Moreno.Trlin@inria.fr (Moreno Trlin), OPalombi@chu-grenoble.fr (Olivier Palombi).

available and if their geometric model is structured with interfaces and elementary functions, e.g. threaded areas of bolts can be used to define FE beam axes, location of bolt holes in plates explicitly identify which holes can be removed when bolts and plates are idealized.

From an analysis of DMUs' content, we describe how their B-Rep model can be processed through a qualitative reasoning process to derive functional information at the interfaces between components up to the global level of components through their functional designation so that meaningful configurations, e.g. assembly joints, can be efficiently identified and processed for FE simulations. Here, the focus is placed on the reasoning process enabling the characterization of functional configurations as well as how geometric models of components get structured from the identification of their geometric interfaces.

The remainder of the paper is organized as follows. In Section 2, we discuss the related work about assembly modeling. Section 3 analyses the content of DMUs and highlights the geometric entities at the basis of the reasoning process. Then, Section 4 introduces the concepts of state, design rules and sets the principles of the qualitative reasoning process. Section 5 focuses on the effective computer aided reasoning process using graphs and ontologies. Finally, Section 6 describes the implementation and illustrates the results currently obtained.

## 2. Related work

Assembly models have been studied and approaches have been proposed throughout the development of solid modeling concepts. Essentially, assembly models have been proposed for design and manufacture applications [5–7] with recent applications to collaborative design. With the development of features, approaches set assemblies as components related to each other through geometric constraints [8]. These approaches share a common denominator where assemblies are described as geometric models enriched with technological up to functional data. However, the closer the functional information, the higher the requirement to obtain external information to CAD environments and the greater the need of user interactive input during a design process. Though these additional informations are mandatory in a design process, it does not appear efficient if incorporated into a preparation process for FE simulation where time reduction is critical to incorporate simulations into a PDP. If they were available at the level of FE simulation, a dedicated approach could be set up to process them but companies are not currently able to set up a digital representation of these data that is connected to the DMUs they produce with CAD systems. Within this framework, Roy and Bharadwaj [9] set up a design approach to connect functions to 3D geometry using a Part Function Model (PFM). There, they address the relationships between function, behavior and geometry of a part in a top-down manner from function to geometry to obtain parts from functional specifications. The PFM described requires up to low level functions that connect part boundary faces to function since the behavior model builds up on interfaces between parts, i.e. contact surfaces to position parts with respect to each other. At the level of complex assemblies with hundreds to thousands of parts, the amount of complementary data defining a behavioral model becomes too tedious to add, and recent approaches [6] reduce the description of junctions between components to global parameters describing bolted and riveted connections without referring to the individual surfaces of each bolt or rivet. This fits well with a design process but simulation objectives may require a detailed representation of interfaces when the purpose is to assess the stress field distribution in a bolted connection with tens or hundreds of bolts.

Developments of ontology-based approaches take advantage of new capabilities to structure concepts and connect them with component models [7,10] or at the level of 3D geometry entities [11–13]. Some of these approaches have been applied to assemblies [7,12,13] and can take advantage of reasoners to set up inference rules to ensure the consistency of the assembly description or extract information that is not readily available in the dataset describing an assembly. In [7,12], there is no direct connection between the ontology content and the 3D model of components. Here, ontologies support a design process where the engineer gets consistent information in a collaborative context but the ontology does not connect to boundary entities of component geometric models. Barbeau et al. [13] cover a larger description of a product with an ontological description incorporating the geometry and structure levels as available through STEP APIs [14,15] up to the functional description that can be inserted through ontological representation of the Core Product Model (CPM) [16]. The authors showed that not all concepts of STEP could be rigorously expressed using OWL standard [17] leading to limitations in detecting inconsistencies. If CPM is able to relate component functions to form features, the content of the CPM reflects the designers' choices to determine the level of product features inserted in this model and its level of connection with each component geometry. Consequently, there is no guarantee that functional information can be available downward to the level of B-Rep entities of each component. Reasoning capabilities associated with ontologies are set up to browse the whole assembly ontology but they are not applied to derive new functional information or reinforce the connection between function and geometry in an assembly model.

Across the tasks involved in FE simulations, specific operators have been set up at the level of FE mesh generation [18–20] to process contact areas between components either from B-Rep CAD models [18] or faceted ones [19,20]. These operators are helpful to reduce the amount of interactive processing to transform component boundaries into geometric models suited for meshing but they are far from the function level to avoid repetitive selection operations and faceted representations can hardly be efficient to pro-

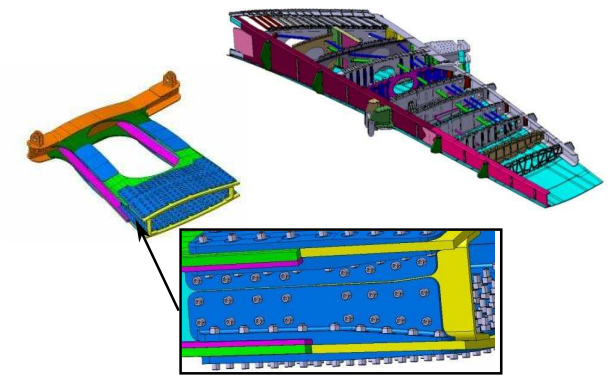

Fig. 1. Two structural sub assemblies of an aircraft wing structure and a detail showing the bolted connections between components (courtesy EADS IW).

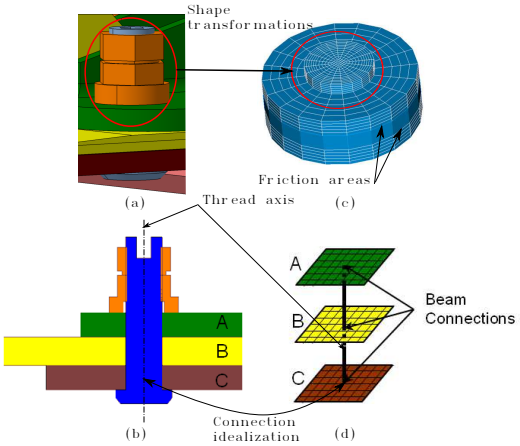


Fig. 2. Example of adaption from DMU to FE analysis model: a) a 3D view of a bolt in a DMU, b) a cutout of a bolt and its neighboring components, c) simplified bolt for detailed stress distribution with friction between plates and discretization with a hexahedral mesh, b) idealized bolt and plates to set up a global model for the bolt as a connector.

cess DMUs during a design process.

Consequently, we can observe that assembly processing is often addressed in a design context of top-down approach from primary product function point of view and there is no low level robust connection between components' geometry and low level functions. Additionally, the reasoning capabilities of ontology-based approaches are exploited to browse existing assembly data and derive data not readily available. These reasoning processes do not extend further than relative positioning constraints between components.

## 3. DMU content, component shapes and conventional interfaces

### 3.1. *DMU content and objectives*

In the industrial context and especially when products addressed are rather complex, DMUs are reduced to a set of components as standalone objects in a common reference frame, where positioning constraints between them are not available. A DMU is an extraction from the PDMS (Product Data Management System) at a given time. Indeed, the evolution of a PDP cannot maintain interfaces between the geometric models of components, e.g. if a component is removed, the resulting loss of consistency in the assembly geometric constraints can propagate through a rather large amount of components. The corresponding geometric constraints have to be removed, hence the choice to locate all components into a global coordinate system. Consequently, each component is positioned independently of the others, which increases the robustness of the DMU with respect to repeated modifications. Figure 1 is an example of a complex structure of an aircraft wing. Mating conditions between components [6,8] are not available.

The hierarchical structure of the assembly as described in a PDMS or in a CAD assembly module is a logical decomposition of the assembly that does not reflect precisely the contact or, more generally, the interfaces between the components. The text-based annotation of each component is not a reliable source of semantic informations attached to components and sub-assemblies. This information may not be stable across a DMU and its content is often irrelevant or rather incomplete with respect to one or more component functions. Indeed, the text-based annotation often contains a code, specific to a company, a project or a product, which does not contribute to a robust connection between a component and its function. Even a `screw`[1] designation does not tell much about its function, i.e. is it a `cap screw`, a `set screw`, an `adjusted cap screw`, ... This text-based annotation does not connect explicitly to some of its faces or edges reflecting the component's functions that rely on the geometric interfaces between the component and its neighbors. Consequently, there is no large difference between a DMU content as obtained from a PDMS and its description as available from a STEP file, which is the approach adopted here as input description of a DMU.

To adapt a DMU to a FE analysis, the engineer conforms to simulation objectives (see examples in section 1) to express shape transformations leading to detail removals and/or idealizations, i.e. dimensional reductions of components. As an example, bolted connections can be simplified to revolution volume models suited for hexahedral meshing (see Fig. 2a, c) or idealized as beam connectors tightening idealized plates (see Fig. 2b, d). To automate such transformations, mandatory information is which components are `cap screws`, `nuts`? which components are tightened by each `cap screw`? where are located the interfaces between `screws` and plates? what is the location of threads' axes? ... Because component interfaces are numerous and close to the physical ones on the real product, the design methodolo-

[1] In order to distinguish clearly the mathematical concept of screw from the real components, all designations of mechanical components will be written using the typewriter typeface, e.g. `screw`.

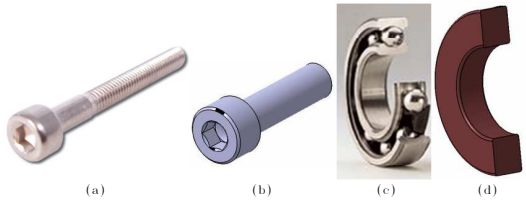


Fig. 3. Examples of components with their real and digital shapes: a) a real `screw`, b) a digital model of a `screw` without the representation of its thread, c) a cutout of a real `ball bearing`, d) a cutout of a digital model of `ball bearing` (all the components of the `bearing` are merged into one ring-shaped volume).

gies relating function, behavior and shape or structure [21–23] can be applied to a small number of interfaces to connect shape to function. It is the purpose of a qualitative reasoning process (see Section 4) to exploit these relationships through component behaviors and answer the previous questions.

### 3.2. *Component shapes and conventional interfaces*

Prior to the presentation of the reasoning process enriching component information, it is important to analyze further the content of DMUs and some principles used to define the shape of components in a design context. The observations of Section 3.1 reduce the robust input data to the geometric model of each component and its location in 3D space. It is now important to observe that the shape of components taking part to a DMU may differ (even significantly) from that of the real object (see Fig. 3). This fact refers to the following definitions.

**Real shape** of a component $C$: it refers to the real physical shape of $C$.

**Digital shape** of $C$: it refers to one possible volume or surface or line model or any of their combinations representing $C$ in a DMU. Here, it is simply designated as a *shape*. Compared to the Real shape of $C$, its shape derives from a simplification process also called idealization.

The selection of a shape for $C$ may be originated by either company standardization or the use of component libraries, e.g. Traceparts (www.traceparts.com), or designer's choices or a combination of some of these reasons, to speed up a modeling process. Consequently, real products contain components that interact with each other through interfaces that can only fall into the categories of: contact (when two component boundaries $\partial C_1$ and $\partial C_2$ touch each other) or functional clearance $\epsilon$ (when subsets of $\partial C_1$ and $\partial C_2$ are located at a distance $d \geq \epsilon$): this is no longer sufficient to analyze DMUs. Indeed, the digital shapes of components can generate interferences in a DMU even though their locations are correct and coincide with the position of their corresponding real component.

As a result, a consistent DMU, i.e. a DMU where all its digital components coincide with the locations of their associated real components, contains digital components that can be related to each other through interfaces. An interface between two components $C_1$ and $C_2$ of a DMU fall into one of the three following categories: contact, interference, clearance, which clearly extends the configurations addressed in most of the existing approaches [8,9,12,7,24]. Here, we assume that processed DMUs are always consistent, hence interferences are acceptable if they exist. Interfaces between components are key information to characterize the context or environment of a digital component. From now on, components designate digital components if there is no ambiguity in its usage context.

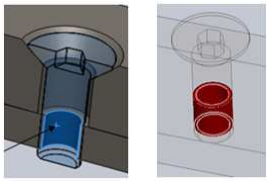


Fig. 4. An example of CI of type interference between a `screw` and an assembly component: a) CI location between the `screw` and the assembly component, b) volume of interference defining their CI.

Though there is no national or international standards stating the digital shapes that ought to be associated to the real shape of $C$, the study of a range of DMUs leads to a conclusion where conventions rather than standards are effectively used at the level of a company or even across companies. These conventions produce invariants. Therefore, we can consider that interfaces between components $C_1$ and $C_2$ are indeed *Conventional Interfaces* (CIs) still belonging to the categories: contact, interference, clearance.

Interfaces between components, hence CIs, contribute to the relative location of these components, to the transmission of forces/moments between them and/or to their relative movements, i.e. what the components are for. CIs are a subset of the result of a product design process, hence they contribute to functions, at a low level. Indeed, CIs contain functional surfaces.

As an example of CI, let us consider a simple, though very frequent, configuration of a `screw` and a `nut` or a component with a threaded hole. Because the shapes of these real components are simplified with the removal of their threaded areas, they just feature cylindrical areas. Now, their diameters are conventionally chosen as the outer diameter of the thread for the `screw` and the drilling hole diameter for the assembly component. The resulting CI is a cylindrical interference forming a ring volume (see Fig. 4).

This DMU analysis shows that design methodologies using DMUs should extend from function, behavior, structure to incorporate the shape of component interfaces as contributors to the relationships between these concepts.

## 4. Principles of the qualitative reasoning process

Design methodologies [21–23] refer to the concept of behavior as a connector between function and shape. Here, the purpose is to define the content and structure of a rea-

soning process, using geometric information as input, and producing functional information as output that is tightly connected to the boundary, i.e. faces, edges, of the components so that the components can be subsequently submitted to shape transformations as required for FE analyses. This reasoning process is set prior to FE analyses with the objective of shortening the FE preparation process, hence a process of qualitative type is well suited since it reduces the amount of required input data.

The process content takes into account the multiple interpretations of interfaces and it is of type bottom-up. As a qualitative process, it can be compared to the reasoning process performed when assembly drawings were analyzed in the past: their analysis was purely conducted without computations and performed with a large amount of geometric information. Consequently, the engineer's reasoning process is based on observations characterizing the relative behavior of components. A DMU analysis is comparable to assembly drawings and we postulate the existence of a qualitative reasoning process. However, the purpose is not to generate a single functional interpretation of an assembly and its components but to derive all the possible and meaningful ones available with the input information given by the user.

An output of the reasoning process holds in the functional designation of a component. It is defined as:

**Functional designation** (FD) of $C$: it refers to a text-based annotation $T_C$ of $C$ such that $T_C$ uniquely identifies the function of $C$, independently of its dimensions and shape. $T_C$ is a member of the taxonomy $\mathcal{T}_{fd}$ associated with the DMU. $\mathcal{T}_{fd}$ contains a collection of FDs of interest for the analysis of this DMU. $T_C$ is connected to the geometric entities of $\partial C$ contributing to the interfaces meaningful for $T_C$ and hence, to the neighboring components of $C$ contributing to $T_C$.

### 4.1. *Conventional Interfaces*

As mentioned in Section 3.1, each DMU component $C_j$ should be in geometric interaction with at least one other component. We call such an interaction a CI (see section 3.2). $C_j$ is a B-Rep CAD volume. Originally, CIs participate to functions to position, transfer forces between, enable relative movements between components. Most of these elementary functions are obtained through surfaces of type plane, cylinder, cone, sphere, torus, collectively designated as $S_f$. Currently, $S_f$ is the set of surfaces that is processed to define the CIs. $S_f$ covers a wide range of interfaces between mechanical components and is readily available from the STEP file containing the DMU.

The geometry processing of a DMU to obtain its CIs is not described in details since the focus is placed here on the reasoning process. Only section 5.1 describes shortly some aspects of the extraction of CIs from a DMU.

A CI can be a *contact*, where the two volumes representing components $C_1$ and $C_2$ share common boundaries, without sharing any volume. A contact is two-dimensional, i.e. a surface contact, or one-dimensional, i.e. a curvilinear contact. More precisely, $C_1$ and $C_2$ can produce different categories of common domain: most often $\partial C_1 \cap \partial C_2 = S$, $S \in S_f$ and $S$ contains only one connected component; if $S$ contains more than one connected component $S_i$ then $\forall S_i$, $S_i \in S_f$ where $S_f$ designates a unique instance of surface, i.e. same type, location in space and intrinsic parameters (see Fig. 5a). Configurations with multiple components belonging to different instances of $S_f$ relate to the configuration of 'double contact' or even 'multiple contacts' (see Fig. 5a), which is inconsistent.

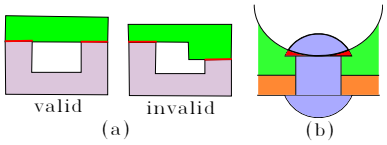


Fig. 5. (a) valid and invalid examples of surface contacts with multiple connected components, (b) Non-manifold interference (in red) obtained as interaction between a `rivet` and two components.

Alternatively, a CI can be an *interference*, where components $C_1$ and $C_2$ share a common domain. Interferences are 3D domains deriving from the relative location of $C_1$ and $C_2$, however, they can exhibit non-manifold configurations, even though the original geometric model of $C_1$ and $C_2$, respectively, is manifold (see Fig. 5b). Here, the taxonomy of CIs is restricted to manifold domains, in a first place.

Clearances are not defined. Indeed, the present reasoning process does not require their extraction from the DMU, which is one of its features.

Whereas contacts often reflect realistic physical configuration, interferences are only idealized representations of reality. As a matter of fact, geometric interactions, either contacts or interferences are no more than conventions that engineers use, and they unevenly reflect reality and how the product should be manufactured. Based on the practices observed in DMUs and an analysis of the content of component libraries, a taxonomy of CIs, $\mathcal{T}_{ci}$, has been set up to cover the basic interactions between components. Fig. 6 illustrates the content of $\mathcal{T}_{ci}$. $\mathcal{T}_{ci}$ does not cover all the possible configurations to shorten its description.

### 4.2. *Functional Interfaces*

Starting from CIs and their interpretations structured into $\mathcal{T}_{ci}$, the next step is to associate at least one or more functional meanings to each class of $\mathcal{T}_{ci}$. The functional meanings that can be assigned to a given CI derives from the conventions adopted during the generation of a DMU. Compared to the real shape of each component, they characterize some shape idealization of components.

The conventional representations used can be stated as:

– Cylindrical fittings. A snug fit is represented with the same nominal diameter for its shaft $C_1$ and housing $C_2$, i.e. its negative clearance is always small enough compared to the nominal diameter of the cylinder to idealize

$C_1$ and $C_2$ to the same diameter. Dimensional tolerances of $C_1$ and $C_2$ are regarded as attributes that may or may not exist in their native CAD models and are regarded as non available. Indeed, they don't exist in the input STEP file. A loose fit with a clearance contributing to a guidance between $C_1$ and $C_2$ is represented with the same nominal diameter. The observation stated for tolerances of the snug fit applies also here. If the clearance between $C_1$ and $C_2$ is such that the two cylinders never touch each other under regular working conditions, there is no guiding function connecting $C_1$ to $C_2$ and their difference in diameter exists in the DMU. The previous observations about their diameter tolerances still holds here,

– Cylindrical interferences. These configurations occur under the following idealizations. Threads are not represented in digital components. They are idealized into a cylinder of diameter equal to the external diameter of the shaft $D_e$ and a cylinder of diameter equal to the drilling diameter of the housing $D_d$. This common convention produces effectively a cylindrical interference $D_e > D_d$, independently of the thread dimensions since this inequality always holds. Spline profiles may or may not be part of the digital shape of $C$ according to the designer's choice and the specific type of spline fit. Throughout this paper, a spline designates a mechanical link. If the real shape of the spline profile appears in the digital shape of $C$, it will produce contact interfaces and, consequently, it does not fall into the current class. If the spline profile is idealized into a simple cylindrical surface, its diameter on the shaft is set to the head diameter of the spline profile $D_e$ and its diameter on the housing is set to the foot diameter of the spline profile $D_f$. This configuration produces a cylindrical interference with $D_e > D_f$, independently of the value of the spline nominal diameter,
– Planar fit. It is always a unique interpretation of planar contact. If $C$ shares two planar CIs, opposite to each other, with other components, the interpretation is part of the reasoning process and does not stand for a class FI. Anyhow, its interpretation should be similar to that of cylindrical fittings,
– Spherical fit. There is always a unique interpretation of spherical loose fit for this CI,
– Conical fittings. Two functional representations are derived from a conical fit that are always based on a conical contact. The distinction between them relates to the adherence phenomenon, which depends on the apex angle of the cone and the adherence coefficient between $C_1$ and $C_2$. This coefficient is obtained from the constitutive materials of $C_1$ and $C_2$ to determine the angle threshold between a loose fit and an adherent fit. Assuming that these materials are unknown, which is generally the case for DMUs acquired through STEP files, both functional interfaces can take part to the reasoning process.

For the sake of conciseness, conventional representations related to curvilinear contacts are not addressed here. As a result, tori have not been addressed since they contribute only to configurations of curvilinear contacts.

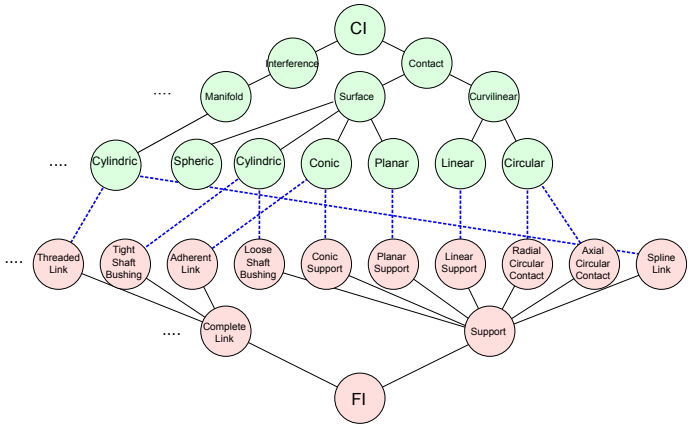


Fig. 6. The taxonomies $\mathcal{T}_{ci}$ of CIs and $\mathcal{T}_{fi}$ of FIs derived from $\mathcal{T}_{ci}$, partially illustrated. $\mathcal{T}_{ci}$ is represented in light blue and $\mathcal{T}_{fi}$ in pink. Dotted lines between the leaves of $\mathcal{T}_{ci}$ and $\mathcal{T}_{fi}$ illustrate the multiple functional interpretations that originate leaves of $\mathcal{T}_{fi}$ from leaves of $\mathcal{T}_{ci}$.

The corresponding collection of FIs ends up with the definition of another taxonomy, $\mathcal{T}_{fi}$. Fig. 6 illustrates the taxonomy $\mathcal{T}_{fi}$ derived from $\mathcal{T}_{ci}$ and the conventional representations of elementary functions. The possible multiple FIs associated with one CI, due to the above conventions, is illustrated in Fig. 6 with the dotted lines connecting leaves of $\mathcal{T}_{ci}$ to leaves of $\mathcal{T}_{fi}$. Because it is hypothesized that DMUs are consistent with the above conventional representations, there must be no interpretation such that an instance of leaf of $\mathcal{T}_{ci}$ connects to more than one instance of a leaf of $\mathcal{T}_{fi}$. Also, a consistent DMU will not exhibit interfaces falling outside the leaves of $\mathcal{T}_{ci}$.

The leaves of $\mathcal{T}_{fi}$, in addition to the geometry of the interfaces obtained from the leaves of $\mathcal{T}_{ci}$ contain the designation of each FI. Starting from this first attachment of functional information, it is now possible to set up behavioral models and design constraints.

### 4.3. *Behaviors*

Now, the FIs need to be processed to filter them out and retain their appropriate interpretation. To this end, the concept of behavior aims at bridging the FIs and the FD of each component $C$. Behaviors can cover a wide spectrum of physical phenomena. In the scope of FE simulations interaction forces between components, i.e. internal forces tightening components among each other are key features that can be exploited to define behavioral models.

If the concepts of behavior and function as addressed in [21,22,25] refer more or less explicitly to state variables, the concept of product or DMU state is not explicitly used as part of a design process. Here, the concept of DMU state is a major concept used to characterize a DMU behavior. Its definition is as follows:

**State** of a DMU: It describes a physical and qualitative behavior of a DMU through equilibrium equations. A behavior law is applied to each component of the DMU where

Table 1
Qualitative vector values.

| Value | Symb. | Interpretation |
|---|---|---|
| `Not Null` | $\not\oslash$ | propagates internal forces / moments in either direction. |
| `Null` | $\oslash$ | doesn't propagate any internal force / moment. |
| `Strictly Positive` | $\oslash_>$ | propagates internal forces / moments in the positive direction only. |
| `Strictly Negative` | $\oslash_<$ | propagates internal forces / moments in the negative direction only. |
| `Arbitrary` | $\circledast$ | may propagate internal forces / moments in either direction |

each interface is assigned a possible FI. This behavior law helps characterize the physical objective of the state and dualities between geometry and mechanics are used to set the parameters of this behavior law from the FIs. Interface shapes and screws describing the interaction forces between components are dual and exemplify these dualities.

The purpose of a state is to filter out FIs to extract consistent functional interpretations so that FIs attached to each $C$ represent an effective possible function of $C$. Applying the behavior law successively to each component until a consistent result is obtained, validates the FIs used in this DMU state. Otherwise, unsuccessful applications of the behavior law reject some candidate FIs of $C$.

A first state contributing to FE analysis, $\mathcal{S}_{se}$, is based on the objective expressing that a DMU, considered in a configuration at 'rest', is such that each of its component must satisfy qualitatively the *static equilibrium* equations, i.e. no component must fall apart in the DMU. Here, the behavior law is the static mechanical equilibrium equations applied to each component.

To perform the corresponding DMU analysis, a screw $\{\mathbf{F}_{C_i/C} \mid \mathbf{M}_{C_i/C}\}$ with qualitative content is assigned to each FI of a component $C$. The content of this screw depends on the meaning of the FI and the duality between interaction forces and the geometry of real interfaces between a couple of components. The equilibrium equation for $C$ writes:

$$\Sigma_i\{\mathbf{F}_{C_i/C} \mid \mathbf{M}_{C_i/C}\} = \mathbf{0}. \tag{1}$$

To process these equations in a qualitative manner, screw components have values among the closed set defined in Table 1 and the equilibrium eq. 1 is expressed using these set of values. Though the multiple FIs attached to each CI produce a combinatorial configuration, advantage can be taken from the design criterion. One of its purposes is to put forward fasteners which are characterized by a small number of FIs. Consequently, eq. 1 is applied, in a first place, to the components having the smallest number of CIs whose FIs contain cylindrical interferences because these interferences generate FIs that can be threaded links, the most common feature of fasteners.

Based on the content of FIs and the qualitative equilibrium equations, it can be observed that a component $C$ containing two CIs; $CI_1$ of type planar contact and $CI_2$ of type cylindrical interference with its axis orthogonal to the plane (see Fig. 7), produces a unique solution of FIs with a

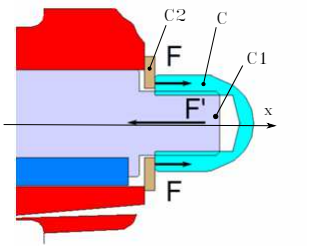


Fig. 7. Static equilibrium of $C$ in a configuration of tightening function.

planar support and a threaded connection. The qualitative screws of the FIs enumerate respectively for the threaded link and the spline fit of $CI_2$:

$$T_{C_1/C} = \{\not\oslash\circledast\circledast \mid \oslash\circledast\circledast\} \text{ and } T_{C_1/C} = \{\oslash\circledast\circledast \mid \circledast\circledast\circledast\} \tag{2}$$

$$T_{C_2/C} = \{\oslash_> \oslash \oslash \mid \oslash \circledast \circledast\} \tag{3}$$

The equilibrium equations written in the reference frame, along its $\mathbf{x}$ axis, containing the axis of $CI_2$ produce respectively:

$$\not\oslash + \oslash_> = \oslash \text{ or } \oslash + \oslash_> = \oslash \tag{4}$$

showing that the FI of type spline fit cannot achieve the static equilibrium of $C$. This configuration discards the spline link FIs to uniquely characterize $C$ with a tightening function. The above example is also a simple illustration of the qualitative operations set up to process the static equilibrium equations since the conciseness of this description prevents from an exhaustive description of the qualitative screws for all FIs and the all set of rules for the addition of screw components.

If $\mathcal{S}_{se}$ can generate some one-to-one mapping between CIs and FIs, as above, it is not always the case and other states are complementary to reduce the number of FIs and improve the functional description of components.

Another state, $\mathcal{S}_{if}$, is also used to filter out some FIs and carry on the characterization of component functions. Considering that threaded links act as *internal force* generators, $\mathcal{S}_{if}$ characterizes the propagation of these forces throughout the DMU. The components of a DMU defining a structure must be subjected to internal forces to be tightened together and hence, avoid any relative movement. $\mathcal{S}_{if}$ aims at defining the FIs belonging to fasteners like `screws` and `rivets` in a DMU that generate internal forces. To this end, the principle of $\mathcal{S}_{if}$ analyzes every FI of type threaded link, $FI_{tl}$, studying qualitatively the static equilibrium of the corresponding component, $C$, starting with components having the smallest number of FIs and threaded links, i.e. two for `screws` or `nuts`. The equilibrium equations of $C$ enables the qualitative determination of the force direction related to $FI_{tl}$, i.e. it replaces $\not\oslash$ either by $\oslash_>$ or $\oslash_<$, and the identification of another FI of $C$, $FI_c$, that sets a force opposite to $FI_{tl}$. Then, this information is propagated to the static equilibrium of the adjacent component sharing the threaded link with $C$. This propagation process is repeated until the force set by $FI_{tl}$ reaches the component $C'$, sharing $FI_c$ with $C$.

At that point, one or several connected closed loops of components in the DMU have been identified that takes part to the tightening function initiated by $FI_{tl}$. The same basic principle can be adapted to other configurations. As an example, it is necessary to process fasteners configurations like studs where two threaded links exist and the initial static equilibrium of a stud, $C$ cannot decide whether the threaded forces produce either a compression or a traction in $C$. In this case, both configurations are initiated to enable the determination of the correct one and the corresponding loops of components.

Given the description of $\mathcal{S}_{if}$, this state is clearly independent from $\mathcal{S}_{se}$, which justifies that the functional information gained through $\mathcal{S}_{if}$ is effectively a complement of $\mathcal{S}_{se}$.

Other behaviors may take place in a DMU, like relative movements of components when a DMU is a mechanism. To carry on reducing the set of FIs attached to each component, a kinematic state has been defined that is independent of the previous states. It represents the DMU in working conditions and its objective is to assign components to kinematic equivalence classes and extract kinematic chains to complement the behavior of components. Relative degrees of freedom are further means to process a larger range of FDs. This is not detailed for sake of conciseness.

### 4.4. *Design constraints*

Behaviors characterized through states express different physical interactions between components and assign them FIs. On a complementary basis, FIs must conform to design constraints. Design constraints refer to dependencies between FIs that contribute to eliminate some FIs. Based on the current taxonomy of FIs, there is no exhaustive list of such constraints yet. Here, the intent is to illustrate some of them that have been identified and relate to the concept of fastener for FE simulations.

A first design constraint, $\mathcal{DC}_1$, addresses the compatibility between 'threaded link', $FI_t$, and 'spline fit', $FI_s$, on a component $C$. Let us express $\mathcal{DC}_1$ when $C$ is a shaft. Here, $FI_t$ and $FI_s$ are co-axial and span an interval $L$ along the axis of $C$. Indeed, these FIs cannot be geometrically located on $C$ and have the same nominal diameter $D$ if they are adjacent, on either of the side of $L$, to a geometric feature of $C$ having a diameter greater or equal to $D$. Fig. 8 illustrates such a configuration. $\mathcal{DC}_1$ expresses the fact that mounting components on $C$ that exploit $FI_t$ and $FI_s$ is not possible and, if $FI_t$ and $FI_s$ are adjacent to each other, the manufacture of $C$ wouldn't be possible. A similar constraint could be set up when $C$ is a housing.

$\mathcal{DC}_1$ is useful when fasteners use functional solutions with nut and counter-nut to discriminate FIs satisfying $\mathcal{S}_{se}$ where the CIs related to the nut and counter-nut can be assigned ($FI_s$, $FI_t$) or two $FI_t$, respectively. Therefore, $\mathcal{DC}_1$ filters out the ($FI_s$, $FI_t$) configuration.

A second design constraint, $\mathcal{DC}_2$, focuses on dependencies between 'threaded link', $FI_t$, 'cylindrical snug fit', $FI_{sf}$, 'cylindrical loose fit', $FI_{lf}$, and 'planar support', $FI_{ps}$, on a component $C$ of type shaft. Here, several $FI_{sf}$ or $FI_{lf}$ exist that are co-linear and adjacent to each other, $FI_t$ is also co-linear and adjacent to $FI_{sf}$ and $FI_{ps}$ is adjacent to $FI_{sf}$ opposite to $FI_t$. $\mathcal{DC}_2$ can be stated as: all the cylindrical fits are of type $FI_{lf}$. $\mathcal{DC}_2$ expresses that the tightening effect of $FI_t$ doesn't need to be reinforced by the $FI_{sf}$ interfaces. To a larger extent, this constraint expresses that a component is designed with elementary functions that are clearly assigned to a single component, i.e. the effect of $FI_t$ is sufficient to tighten all the components attached to $FI_{lf}$.

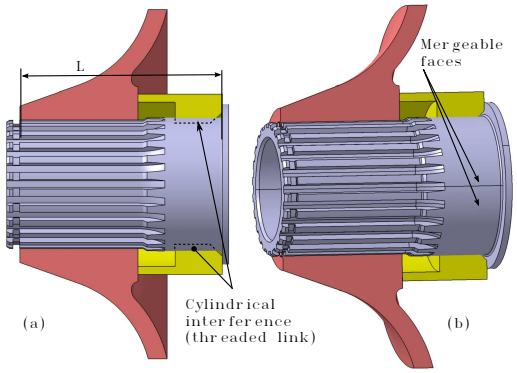


Fig. 8. An example of layout of FIs falling in the scope of a design constraint. The spline fit is represented realistically for convenience purposes. a) a cutout highlighting the incompatible location of the cylindrical interference with respect to the spline link; b) examples of faces that can be merged to produce maximal faces. The merged configuration is depicted in (a).

$\mathcal{DC}_2$ is useful when fasteners are of type `adjusted bolts` where `screws` are mounted with loose fit.

As illustrated above, design constraints form a necessary complement to the behaviors expressed by states. The identification of design constraints is still at a preliminary stage. It can be observed that the dependencies between FIs expressed by design constraints inherit also from the CIs, hence their dependency with respect to the engineer's representation choices of interfaces in DMUs.

### 4.5. *Functional Designations of components*

The concept of FD has already been introduced at the beginning of Section 4. To be able to assign a FD to each component, a taxonomy of FDs, $\mathcal{T}_{fd}$ is required that can connect all the FIs of a component to a unique functional concept that can characterize all the components, i.e. instances, sharing the same function.

$\mathcal{T}_{fd}$ should cover all the categories of functions of components in a DMU. Currently, it has been restricted to fasteners, which means that only a subset of components in a DMU can be assigned FDs. Though $\mathcal{T}_{fd}$ is partial only, there may be influences of other categories of components, not described in $\mathcal{T}_{fd}$, on the correct assignment of FDs for the components that would belong to these categories but are indeed assigned to one of the classes in $\mathcal{T}_{fd}$ because


Fig. 9. A partial description of the structure of $\mathcal{T}_{fd}$.

these components have common properties with classes in $\mathcal{T}_{fi}$ and the existing states and design constraints are not able to separate these components from the classes in $\mathcal{T}_{fd}$.

Given the above remark, it can be observed that the behaviors and design constraints described at Sections 4.3 and 4.4 are necessary conditions to correctly identify the classes of components in $\mathcal{T}_{fd}$ but they are not sufficient.

The partial content of $\mathcal{T}_{fd}$ is illustrated in Fig. 9. It can be noticed that components having assigned a class in $\mathcal{T}_{fd}$ benefit from an entire functional description with interfaces having functions assigned, these functions being clearly identified through faces, edges and vertices on the boundary of these components. In addition, these components having been processed through states, some of their behaviors are now available, which can be used for FE simulation preparation to apply shape transformations and generate meshes as efficiently as possible.

## 5. Computer aided qualitative reasoning process

Section 4 has described the major concepts of the proposed approach to analyze DMUs and derive FDs of components from their CIs. These concepts have been stated independently of any category of reasoning process or of type of mathematical logic. Here, the purpose is to point out some of the issues addressed to effectively structure the reasoning process. Depending on the category of reasoning process, existing software tools are used as much as possible like ontologies [26,17] and their associated reasoners [27] because qualitative approaches and functional information are well suited to use logics.

In this section, we highlight some technical aspects of our method at some of its elementary phases. Figure 10 depicts different stages of our approach. The input is a pure geometric description of a product as STEP [14,15] file containing a B-Rep model.

### 5.1. *Maximal edges/surfaces and geometric interfaces detection*

B-Rep CAD models don't represent product's shape uniquely; that means the same volume can have several different valid representations under B-Rep modeling. This originates partly from the design process where any face can be subdivided into smaller ones during a design

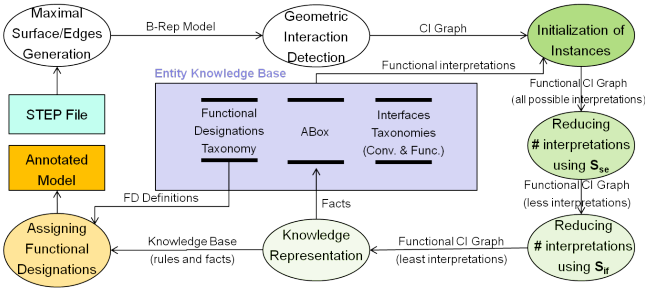


Fig. 10. Data and process flow diagram of the reasoning process.

process and partly from modeling constraints set by B-Rep modelers. Fig. 8b illustrates these two origins with faces referenced on the shaft of the assembly. The same phenomenon applies to edges also. To overcome this inconvenience, we first generate the so-called *maximal surfaces* and *maximal edges* representation of DMU components. For each component, this is accomplished by merging all adjacent surfaces and curves that share the same geometric properties, i.e. type, 3D location, intrinsic parameters (Fig. 8a illustrates a result of this process). This is mandatory to correctly define the interfaces and detect some inconsistencies (see Fig. 5a).

Having generated the maximal faces and edges, it makes way for the determination of geometric interactions between volumes that define CIs. A naive approach for the detection of geometric interfaces is to use boolean operators between volume pairs of the DMU at hand. This can be enhanced by using the elimination of obvious negative pairs whose bounding boxes do not interact with each other. However, not only interferences but also surface and line contacts are needed but boolean operators don't bring the right level of robustness and efficiency, as it has been evaluated using the OpenCascade CAD library [28]. This technique however fails short to deliver accurate interface zones. This shortage requires specific approaches but they are not described here for sake of paper length.

The outcome of this phase is represented as a non-oriented binary graph referred to as *Conventional Interface Graph* (CIG): $G_{CI} = (N_c, A_i)$. Here, nodes $N_c$ are DMU components and arcs $A_i$ are interfaces. This is a mathematical model upon which we build our reasoning phases to come. Initially, graph arcs, i.e. CIs, contain information about only the geometric interaction between two nodes. They define instances populating $\mathcal{T}_{ci}$.

### 5.2. *Initialization of knowledge base and reducing the number of interpretations*

CIs are then enriched by functional interpretations to produce FIs and conform to the connections between $\mathcal{T}_{ci}$ and $\mathcal{T}_{fi}$ (see Fig. 6). Thus, one entry of $\mathcal{T}_{ci}$ may lead to more than one instance of $\mathcal{T}_{fi}$. This is due to the idealized nature of some CIs (see Section 4.2).

$G_{CI}$, now enriched with instances of FIs located at $A_i$, reflects our initial knowledge about the assembly. Uncer-

tainty in the knowledge base comes from the fact that some CIs still hold more than one FI, thus, all the FIs of a component $C$ can't yet contribute to instantiation of $C$'s FD. To reduce such uncertainty, additional rules or/and facts, i.e. behaviors and design constraints, are considered.

States $\mathcal{S}_{se}$, $\mathcal{S}_{if}$ and design constraints, $\mathcal{DC}_1$, $\mathcal{DC}_2$, come to clarify ambiguities by reducing the number of interpretations per CI to ideally one. These states are hypotheses considered to hold true during a specific stage and express an elementary behavior of some DMU components. Those hypotheses are formed as rules that add up to our knowledge base and contribute to the elimination of FIs that contradict one or more of those rules.

Two reference states have so far been identified and set up, $\mathcal{S}_{se}$, $\mathcal{S}_{if}$. They share a similar use of qualitative static mechanical analysis of DMU components hypothesized as rigid bodies. However, the basic vector algebra required to process the qualitative screws cannot be handled by reasoners connected to ontologies. Reducing the vector equations to scalar ones at the level of eq. 1 can help reach the compatibility with reasoners but processing large assemblies that way may not be efficient as highlighted in [13] with the limitations faced in expressing some STEP concepts and poor performances that may occur. For these reasons, dedicated algorithms have been set up to process $\mathcal{S}_{se}$, $\mathcal{S}_{if}$.

Each FI has a qualitative mechanical screw expressed in a reference coordinate system specific to this FI. Since all screws involved in the equilibrium of a component $C$ must be expressed in the same reference frame, these screws must be subjected to vector field transformations, i.e.

$$\begin{aligned} \{\mathbf{F}_{O_C}|\mathbf{M}_{O_C}\}_{O_C} = \{\mathbf{F}_{O_F}|\mathbf{M}_{O_F}\}_{O_F} \implies \\ \mathbf{F}_{O_C} = \mathbf{F}_{O_F} \text{ and } \mathbf{M}_{O_C} = \mathbf{M}_{O_F} + \mathbf{O}_C\mathbf{O}_F \times \mathbf{F}_{O_F} \end{aligned} \tag{5}$$

where $O_F$ and $O_C$ designate respectively the origin of the reference frame of an FI and the origin of the reference frame of $C$ set up for all FIs. After this transformation, the qualitative screws of FIs are referred to as *absolute screws*.

Special arithmetics have also been defined to allow operations such as addition, subtraction, scaling (scalar multiplication), dot product, ... between vectors with qualitative component values.

An exhaustive search algorithm is presented in algo. 1. It is based on $\mathcal{S}_{se}$ and returns all valid solutions to eq. 1, eliminating invalid interpretations for each FI.

This algorithm traverses the $G_{CI}$ nodes, visiting each node at least once, to study the component's equilibrium against eq. 1. All nodes of $G_{CI}$ are initially marked as open, and thus still to be visited. Method `next_open_component()` chooses the component with minimum cardinality, defined as the product of number of interpretations over CIs of a component $c$: $||c|| = \prod_{ci \in CI_c} |ci|$ where $|ci|$ is the number of interpretations of $ci$. If two components happen to have the same cardinality the one with fewer interfaces is chosen. This heuristic helps picking components that have the potential to introduce more information to the knowledge base than others, enhancing the reasoning performance.

**Algorithm 1** Mechanical analysis

```
Procedure: analyse
  for each component c do
    mark c as open
  end for
  while there is still open component do
    c ← next_open_component
    init_screw ← {0|0}
    initialize all interpretations as invalid
    calculate_sum(c, 0, 0, init_screw)
    for interpretations that are still invalid interpret do
      interface ← interpret's CI
      other ← interface's opposite component
      mark other as open
      drop interpret
    end for
    if c's cardinality didn't change or c's cardinality = 0 then
      mark c as closed
    end if
  end while
Procedure: calculate_sum(c, level, i, base)
  if is_arbitrary(base) = true then
    mark all visited interpretations as valid
    mark all interpretations to be visited from here as valid
  else
    if level = number of interpretations then
      if is_nullable(base) = true then
        mark all visited interpretations as valid
      end if
    else
      interface ← level-th interface of c
      for i = 0 to number of interpretations of interface do
        interpret ← i-th interpretation of interface
        screw ← base + interpret.screw
        calculate_sum(c, level + 1, i, screw)
      end for
    end if
  end if
```

Component's equilibrium is studied through method `calculate_sum()`. If this study leads to the elimination of some FIs, not only the state of the current component stays open, also the opposite component of the eliminated FI is marked open as well, even if it was closed before. This is because the removal of one FI introduces new knowledge that may in turn allow for more conclusion if equilibrium against the components is questioned again. If no interpretation is removed, that means that further reasoning on the component is meaningless, since it will lead to the same result, then it is closed.

The method `calculate_sum()` traverses all CIs of a component recursively, through a depth first graph search, combining solutions and checking the validity of each against eq. 1 at leaf level. This is done through the accumulation of mechanical screws, where the screw of the currently visited FI is added to a sum which is eventually checked whether or not to be *nullable*; that is, whether or not eq. 1 may hold true. If the answer is positive, all visited FIs are marked as valid. To this end, a track of visited FIs is kept, though it is not mentioned in the outlines of the algorithm.

An enhancement is introduced by the early determination of valid solutions when a screw represents a complete link, determined by method `is_arbitrary()`. In this case, summation will always lead to a nullable screw value, thus the recursion is interrupted, and all solutions still to be discovered from this point are marked as valid.

Algo. 1 allows reasoning upon FIs. Results obtained at this stage adds up to the ABox of our knowledge base, al-

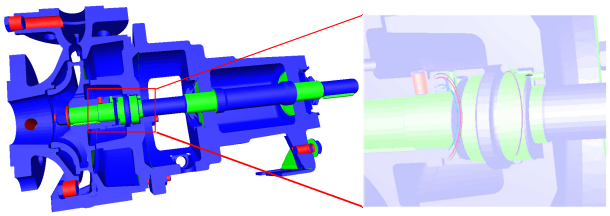

Fig. 11. CIs of a DMU of hydraulic pump with 72 components.

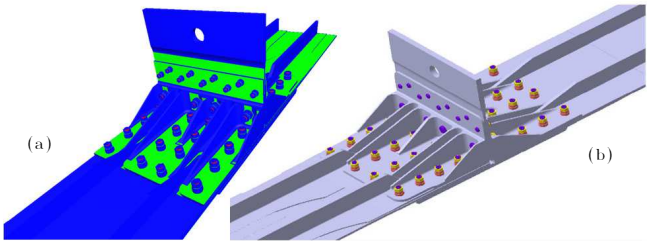


Fig. 12. A subset of a connection of a wing structure to a fuselage. a)interfaces forming $G_{CI}$, b) FDs of the components.

ready containing $\mathcal{T}_{ci}$, $\mathcal{T}_{fi}$, and $\mathcal{T}_{fd}$ in its TBox. The knowledge base is represented as an ontology using OWL [17]. To classify components into $\mathcal{T}_{fd}$, further reasoning should be made. To this end, rules are set up that uniquely distinguish a class of components, and assign it an FD. Let us exemplify the definition of a `screw`, shown in rule 1.

**Rule 1** *Any component that forms the inner part its $FI_t$, and has a $FI_{ps}$, such as the axis of $FI_t$ is orthogonal to $FI_{ps}$, are said to be a* `screw`.

This rule is expressed using a Description Logic (DL) and reasoning using reasoners such as FaCT++[27].

## 6. Implementation and results

Here, we mention certain implementation details, then we demonstrate results of our automated process.

### 6.1. *Technical choices*

For geometrical analysis of DMUs we used OpenCascade[28] platform, that allows for the reading and extraction of geometrical entities and their properties out of a STEP file. The knowledge base is set up using Protégé[26] in connection to MyCorporisFabrica [29] to express our rules and communicate with FaCT++ through its DIG interface.

### 6.2. *Results*

Fig. 11 shows the diversity of interfaces extracted from a DMU of a hydraulic pump. Red areas indicate interferences, green ones are surface contacts on Fig. 11. Line contacts are indicated on Fig. 11(right) as a detail. Blue areas have no status with regard to interfaces.

Fig. 12 illustrates a subset of an aircraft wing structure with 104 components. Fig. 12a indicates the geometric interfaces obtained from the DMU to form $G_{CI}$. The color code here is identical to Fig. 11. Fig. 12b shows the results of the DMU analysis using the reference states, design constraints, and semantic reasoning. Colored components (rather than default gray) are identified fasteners. Violet components are `screws`, while yellow ones are `counter-nuts` and pink ones are `nuts`.

Fig. 13 shows a zoom into a cross section in the same DMU of Fig. 12, along with the corresponding subgraph of $G_{CI}$. The graph shown depicts how validation against $S_{se}$

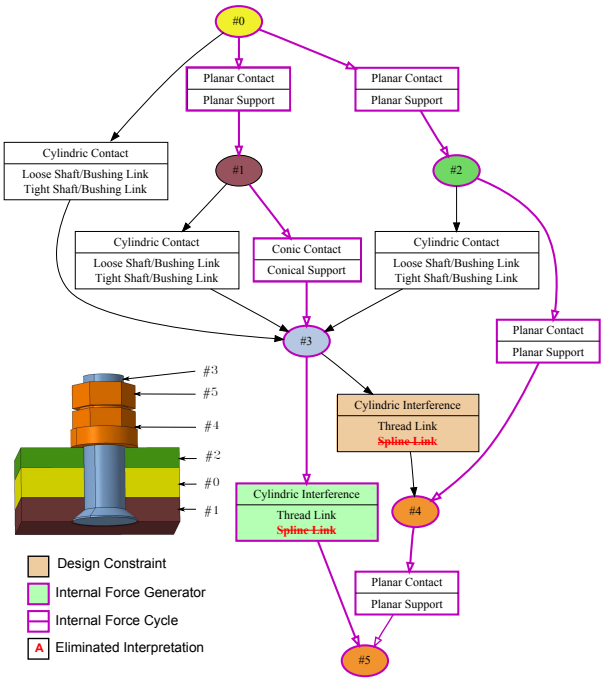


Fig. 13. Result of validation against reference states $S_{se}$ and $S_{if}$.

enable the removal of interpretations such as Spline Link between components #3 and #5, as they invalidate static equilibrium. Further interpretations are eliminated thanks to $\mathcal{DC}_1$, such as Spline Link between components #3 and #4. Reasoning upon only $S_{se}$ would not allow for such rejection. Validation against $S_{if}$ shows how cycles that propagate internal forces are reported, such a cycle is shown as magenta-colored edges in the same figure. This example demonstrates that components can be functionally identified. Threaded areas can be located on components to initiate idealizations, and fasteners are associated to the components they tighten and to their interfaces. It is then possible to use this information to set up friction areas around fasteners as needed for some FE models.

## 7. Conclusion and future work

It has been analyzed how component shapes, in assemblies, contribute to the definition of interfaces with their neighbors. These geometric interfaces, CIs, can be connected to functional interfaces: interpretations of these FIs through conventional representations of components estab-

lish a connection between component shape and low level functions of their interfaces. A CI gives birth to one or more FIs. Though CIs and FIs can be structured into taxonomies, it has been demonstrated that complementary functional information can be gained through the notion of behavior addressed here qualitatively with the concept of DMU states that can filter out FIs and provide higher level functional information. As a complement to states, design constraints taking care of dependencies between FIs appeared also as a key concept to attach meaningful FIs to each component so that it can be identified functionally by a denomination out of a functional designation taxonomy. This FD becomes a key information to apply specific shape transformation operators to derive simulation models for FE analyses.

The proposed approach, through a qualitative reasoning process, takes advantage of ontology-based representations and their associated reasoning capabilities. However, specific stages of the reasoning process cannot be expressed with reasoners' logic and need to be set as algorithms. The reasoning process builds upon the shape, function, behavior design methodologies with the concepts of CIs, FIs, FDs, states and design constraints, in bottom-up manner to attach functional information to DMU components.

Future work will extend the present approach with new states and further investigations of the relationships between shapes, interfaces, states, design constraints and functions.

## Acknowledgments

This work is carried out in the framework of the ANR project ROMMA (RObust Mechanical Models for Assemblies) referenced ANR-09-COSI-012 and the authors thank the ANR for its financial support. This work is also supported by the ERC Expressive.

## References

[1] A. Thakur, A. G. Banerjee, S. K. Gupta, A survey of cad model simplification techniques for physics-based simulation applications, CAD 41 (2) (2009) 65–80.

[2] C. S. Chong, A. S. Kumar, K. H. Lee, Automatic solid decomposition and reduction for non-manifold geometric model generation, CAD 36 (13) (2004) 1357–1369.

[3] T. T. Robinson, C. G. Armstrong, R. Fairey, Automated mixed dimensional modelling from 2d and 3d cad models, Finite Elements in Analysis and Design 47 (2) (2011) 151–165.

[4] C. Eckard, Advantages and disavantadges of fem analysis in an early state of the design process, in: Proc. $2^{nd}$ Worldwide Automotive Conf., MSC Software Corp., Dearborn, Michigan, USA, 2000.

[5] U. Roy, N. Pramanik, R. Sudarsan, R. D. Sriram, K. W. Lyons, Function–to–form mapping: model, representation and applications in design synthesis, CAD 33 (10) (2001) 699–719.

[6] K.-Y. Kim, Y. Wang, O. S. Muogboh, B. O. Nnaji, Design formalism for collaborative assembly design, CAD 36 (9) (2004) 849–871.

[7] K. Rahmani, V. Thomson, Ontology based interface design and control methodology for collaborative product development, CAD 44 (5) (2012) 432–444.

[8] W. van Holland, W. F. Bronsvoort, Assembly features in modeling and planning, RCIM 16 (2000) 277–294.

[9] U. Roy, B. Bharadwaj, Design with part behaviors: behavior model, representation and applications, CAD 34 (9) (2002) 613–636.

[10] Y. Kitamura, R. Mizoguchi, Ontology-based systematization of functional knowledge, J Eng. Design 15 (4) (2004) 327–351.

[11] I. Horvath, J. P. W. Pulles, A. P. Bremer, J. S. M. Vergeest, Towards an ontology-based definition of design features, in: Proc. SIAM Workshop on mathematical foundations for features in computer aided design, engineering, and manufacturing, 1998.

[12] K.-Y. Kim, D. G. Manley, H. Yang, Ontology-based assembly design and information sharing for collaborative product development, CAD 38 (12) (2006) 1233–1250.

[13] R. Barbau, S. Krima, S. Rachuri, A. Narayanan, X. Fiorentini, S. Foufou, R. D. Sriram, Ontostep: Enriching product model data using ontologies, CAD 44 (6) (2012) 575–590.

[14] I. TC184-SC4, ISO-10303 Part 203 - Application Protocol: Configuration controlled 3D design of mechanical parts and assemblies, ISO, 1994.

[15] I. TC184-SC4, ISO-10303 Part 214 - Application protocol: Core data for automotive mechanical design processes, ISO, 2003.

[16] X. Fiorentini, I. Gambino, V.-C. Liang, S. Rachuri, M. Mani, C. Bock, An ontology for assembly representation, Tech. rep., National Institute of Standards and Technology NISTIR 7436, Gaithersburg, MD 20899, USA, July, (2007).

[17] OWL Web Ontology Language Overview (2004).
URL `http://www.w3.org/TR/owl-features/`

[18] B. Clark, B. Hanks, C. Ernst, Conformal assembly meshing with tolerant imprinting, in: Proc. 17th Meshing Roundtable, Pittsburg, USA, October 12–15, 2008, pp. 267–280.

[19] R. Chouadria, P. Véron, Identifying and re-meshing contact interfaces in a polyhedral assembly for digital mock-up, Eng. with Computers 22 (1) (2006) 47–58.

[20] R. Lou, J.-P. Pernot, A. Mikchevitch, P. Véron, Merging enriched finite element triangle meshes for fast prototyping of alternate solutions in the context of industrial maintenance, CAD 42 (8) (2010) 670–681.

[21] J. S. Gero, U. Kannengiesser, The situated function–behaviour, Design Studies 25 (2004) 373–391.

[22] A. Albers, N. Burkardt, M. Ohmer, Contact and Channel Model for Pairs of Working Surfaces, In ElMaraghy, H. A., ElMaraghy, W. H. (Eds), London: Springer, 2006, Ch. Advances in Design, pp. 511–520,.

[23] D. G. Ullman, The mechanical design process, 2nd ed. New York: McGraw-Hill, 1997.

[24] N. J. Mitra, Y.-L. Yang, D.-M. Yan, W. Li, M. Agrawala, Illustrating how mechanical assemblies work, in: ACM SIGGRAPH, 2010.

[25] B. Chandrasekaran, J. R. Josephson, Function in device representation, Engineering with Computers 16 (2000) 162–177.

[26] Protégé (2012).
URL `http://protege.stanford.edu/`

[27] D. Tsarkov, I. Horrocks, Fact++ description logic reasoner: System description, in: Int. Joint Conf. on Automated Reasoning (IJCAR 2006), vol 4130 of LNCS, 2006, 292–297.

[28] OpenCascade CAD software library (2012).
URL `http://www.opencascade.org/`

[29] O. Palombi, G. Bousquet, D. Jospin, S. Hassan, L. Revéret, F. Faure, My corporis fabrica: a unified ontological, geometrical and mechanical view of human anatomy, in: Lecture Notes in Computer Science LNCS 5903, 2009, pp. 207–219.